REMARKS

The Examiner is thanked for his Office Action.

The claims presently outstanding are Claims 1-7. By the foregoing amendments, various Claims are sought to be amended. These changes are believed not to introduce new matter, and their entry is respectfully requested.

The foregoing amendments to the specification are submitted to improve clarity, and to remove various typographical and other minor informalities. These changes are respectfully asserted not to introduce new matter, and their entry is respectfully requested. The specification has been very extensively reviewed and checked against the drawings. Note that reference 429 in Figure 4B was simply misdescribed (as 427) on page 50, and that typo has now been corrected. Note also that subcomponent 451 of the VDA 445 in Figure 1, which was not described in the text, has simply been deleted. Also note that reference 427 was already present in Figure 1. Thus all objections to drawings and references are believed to have been accommodated (and are traversed).

The Specification has been changed correspondingly. The text added to the Specifications tracks the as-filed drawings very closely, and hence is believed not to introduce new matter.

Section 112(2) ("Clearly and Distinctly Setting Forth")

The undersigned attorney is eager to remove any informalities, but is not clear how to respond to the outstanding 112(2) rejections without inappropriately narrowing the claims. **A telephone conference to discuss this point is requested.** In that telephone conference, the undersigned attorney would like to discuss Figure 8's four illustrated assignments, as an example of support for the claim language in question.

Thus these rejections are traversed, but the undersigned attorney would welcome suggestions for amending language which might be clearer without sacrificing scope.

Applicant specifically notes that Claim 1 is not intended to specify what is mapped to what, and is sufficiently clear to inform those of ordinary skill what the applicant regards as his invention.

It is also noted that the limitation on lookup tag length in Claim 7 has been clarified, and also that the claim text has been copied into the specification. These changes are believed to remove any unclarity.

The Examiner's rejections under §112(1) are respectfully traversed, but are believed to be moot in view of the requested amendment which bodily copies the text of the independent claims into the Specification.

Conclusion

Thus, all grounds of rejection and/or objection are traversed or accommodated, and favorable reconsideration and allowance are respectfully requested. The Examiner is requested to telephone the undersigned attorney for an interview to resolve any remaining issues.

Respectfully submitted,

Robert Groover, Reg.No. 30,059
Attorney for Applicant

Arter & Hadden LLP
1717 Main St. No.4100
Dallas TX 75201
(214) 761-4410

April 2, 2003

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
Appl. No. 09/591,532

Version with Markings to Show Changes Made

In the Specification:

Please rewrite the heading at line 2 of page 1 as follows:

Background [and] [Summary] of the Invention

Please rewrite the paragraph beginning at line 13 of page 2 as follows:

Figure 2 shows a high-level overview of the processes performed in the overall 3D graphics pipeline. In **step 202**, 3D world coordinates are transformed into view coordinates within the canonical view volume. In **step 204**, clipping is performed against the canonical view volume. In **step 206**, the resultant data is projected onto the view plane, and then (**step 208**) mapped into the view port (and into normalized device coordinates). In **step 210** this data is transformed to physical device coordinates, and then (**step 212**) rendered. However, this is a very general overview, which ignores the crucial issues of what hardware performs which operations.

Please rewrite the heading at line 29 of page 8 as follows:

Summary of the Inventions: Direct-Mapped Texture Caching with Concise Tags

Please rewrite the paragraph beginning at line 7 of page 9 as follows:

Notable (and separately innovative) features of the texture caching architecture described in the present application include at least the following: Expedited loading of texel data (preloading, not just prefetching); an improved definition of keys (rather than addresses) for Cache lookup; and an innovative cache replacement policy.

Please rewrite the consecutive paragraphs beginning at the last line of page 13, and ending on line 10 of page 14, as follows:

Figure 3 shows a block diagram of a graphics processor which can incorporate the disclosed embodiments of the read-modify-write solutions in its rendering subsystem. A sample board 300 incorporating the P3™ graphics processor may include these elements:

the P3™ graphics core 310 itself, including rendering subsystem 310A;

a PCI/AGP interface 304;

DMA controllers 340A/340B for PCI/AGP interface to the graphics core and memory respectively;

SGRAM/SDRAM 350, to which the chip 310 has read-write access through its frame buffer (FB) and local buffer (LB) ports 302;

a RAMDAC 320, which provides analog color values in accordance with the color values read out from the SGRAM/SDRAM 350; and

a video stream interface 306 for output and display connectivity.

Please rewrite the consecutive paragraphs beginning at line 29 of page 18 as follows:

For Patch64 the 2D ij coordinate space **500A** is mapped to a 1D address range 500B as shown in **Figure 5**, in which Pixel Offset (top left origin) is given by:

```
i % 64 +                        // i within a patch
(i / 64) * 1024 +               // i between patches
(j % 16) * 64 +                 // j within a patch
(j / 16) * width * 16           // j between patches
```

This can be converted into a simpler calculation just using shifts and adds:

```
(i & 0x3f) + ((i & 0xffc0) << 4) +
((j & 0xf) << 6) + ((j & 0xfff0) * width).
```

For bottom left origin the equation is:

```
(i & 0x3f) + ((i & 0xffc0) << 4) -
((j & 0xf) << 6) - ((j & 0xfff0) * width).
```

Note that corresponding memory pages **502** (of 1K words) are shown both in the 1D layout 500B and in the 2D layout 500A.

Please rewrite the paragraph beginning at line 32 of page 21 as follows:

The patch (2x2, etc.) has a fixed relationship to the origin of the texture map such that the origin of the patch is always some integral multiple of the patch size from the origin of the texture map. Figure 6 [The] [following] [diagram] shows the 2x2 patch arrangement within a texture map 600. The numbers in the brackets show how the texel coordinates **604** within the texture map vary, and the T0...T3 are the corresponding filter registers **602** which each texel is assigned. The grey areas [are] show the texels held in a memory word (16 bytes) for each size of texel - 4 for 32-bit texels (area 610), 8 for 16-bit texels (area 612), or 16 for 8-bit texels (area 614). The texture map may also be patched at a higher level (32x32) to reduce the effect of page breaks but this is of no consequence to how the primary cache functions[(see Figure 6)].

Please rewrite the paragraph beginning at line 27 of page 22 as follows:

The layout in memory for the various supported [format] formats is shown in **Figures 7A-7B**. Each line is one memory word and the bit numbers are shown along the top. The tick marks are at byte intervals and the numbers in brackets show how the texel coordinates vary within the memory word. For linear or Patch64 layouts, example layouts 702, 704, and 706 are shown (depending respectively on whether the texel size is 32 bits, 16 bits, or 8 bits). For Patch32_2 or Patch2 layouts, example layouts 712, 714, and 716 are shown (depending respectively on whether the texel size is 32 bits, 16 bits, or 8 bits).

Please rewrite the paragraph beginning at line 32 of page 24 as follows:

The key (as already described) holds the i and j index and the map level (3D textures will be considered shortly). The maximum width and height of a map is 2050 (2K + a border) so the indices have 12 bits. The cache line holds a 2x2 patch so the indices can be reduced by one bit to 11 bits. The number of map level is needed here. In total the key is (11 + 11 + 4) bits or 26. This can be reduced down to 23 by realizing that the full 2050x2050 value can only occur on map level 0 (as shown in assignment 802 of Figure 8). Map level 1 has a maximum size

of 1026x1026, as shown in assignment 804, and map level 2 needs even fewer i and j bits, as shown in assignment 806. As shown in assignment 808, for levels 3 and higher there is room to include a three-bit map level value as well as 8 bits each for i and j. Thus [so] by encoding the map into the upper bits as shown in **Figure 8**, the key width can be reduced.

Please rewrite the paragraph beginning at line 10 of page 26 as follows:

A fragment could cause from one to eight memory reads, although if the cache is working well and scanline coherency is being made use of this will very much reduced. (The pathological case is where bilinear filtering is being done with a zoom ratio of 1:*n*, where $n > 1$. In this case we are minifying the map and no coherence between adjacent fragments or scanlines can be exploited. From 1 to 4 reads per fragment are needed depending on how the sample points interact with the underlying 2x2 patch structure in the texture map.) **Figure 9** shows which texels **902** the memory reads bring in, and the corresponding output fragments **906** they will satisfy. The zoom ratio of 1:1 is used as this is the worst case for mip mapping and occurs for the higher resolution map; the lower resolution map will have a zoom ratio of 2:1 so any results for this map level will be twice as good. A texel size of 32 bits is also assumed (four texels 902 per word **904**), so these results are independent of any path orientation. The smaller texels sizes will give better results for X major paths.

Please rewrite the paragraph beginning at line 8 of page 47 as follows:

In servicing the interrupt a physical page (or pages if the interrupt is used to allocate a whole texture rather than just a page) must be allocated by software. If these physical pages are already assigned then the corresponding logical pages must be marked as nonresident [non] [resident] in the Logical Texture Page Table. If these newly nonresident [non] [resident] logical pages are subsequently accessed (maybe by a queued texture operation) they themselves will cause a page fault and be reassigned. [re] [assigned.] Hence no knowledge of what textures are waiting in the DMA buffer to be used is necessary. The physical pages

are allocated from the host working set whose base page is given by BaseOfWorkingSetHost register.

Please rewrite the paragraph beginning at line 29 of page 50 as follows:

Figure 4B shows actions in the Primary Cache Manager. If a cache miss occurs (test 421), the details of the missing texel are obtained (step 423), and the next free cache line is looked up (step 425). A read command is then issued to the address generator (step 429), ~~[427),]~~ specifying the free cache line as the return address. The address generator updates the T FIFO after the read request has occurred. A message is then written into the M FIFO with details of the cache lines used, fragment details, and the number (if any) of additional cache loads which have now occurred.

Please rewrite the paragraph beginning at line 29 of page 50 as follows:

Figure 4B shows actions in the Primary Cache Manager. If a cache miss occurs (test 421), the details of the missing texel are obtained (step 423), and the next free cache line is looked up (step 425). A read command is then issued to the address generator (step 429), ~~[427),]~~ specifying the free cache line as the return address. The address generator updates the T FIFO after the read request has occurred. A message is then written into the M FIFO with details of the cache lines used, fragment details, and the number (if any) of additional cache loads which have now occurred.

Please rewrite the sequential paragraphs beginning at line 18 of page 51 as follows:

The Primary Cache Manager **1070**, Address Generator **1060** and Dispatcher 1080 form the core of the unit and work in a similar way to the other read units. The logical address translation is handled by the Address Mapper 1050 and TLB 1040. The dynamic texture loading is handled by the Memory Allocater 1030 and the Download Controller 1020.

The interfaces between all the units are shown as FIFOs 1090, but

most of the FIFOs 1090 are just a register with full/empty flags for simple handshaking. The single deep FIFOs 1090 have been used as they clearly delineate the functionality between units and allow a single sub unit to be responsible for a single resource.

The two shared resources which are managed in this way are the TLB 1040 and Memory Allocater 1030. The TLB is mainly queried by the Address Mapper 1050 but the Memory Allocater 1030 needs to invalidate pages when a physical page is re-assigned. The Memory Allocater 1030 will allocate pages when requested by the Download Controller 1020, but also needs to mark pages as "most recently used" when requested by the Address Mapper 1050.

There are two read/write ports to the Memory Controller 1010 used to access the Logical Page Table and the Physical Page Allocation Table - these are 64 bit ports and are not FIFO buffered. There is no point in trying to queue up reads or writes on these ports as the texture process stalls until these operations are satisfied.

The read port to the Memory Controller 1010 is used to read texture data and has a deep address FIFO and return data FIFO to absorb latency.

The write port to the Memory Controller 1010 is used by the Download Controller 1020 to write texture data into memory during a download. The path from the Texture Input FIFO to the Memory Controller 1010 is 128 bits wide so the maximum download bandwidth can be sustained.

All the controlling registers (TextureReadMode, TextureMapWidth, TextureBaseAddr, etc. are all held in the Primary Cache Manager 1070 so the responsibility for loading them from the message stream, context dumping and readback is all concentrated in one place. This does mean that before any of them can be updated any outstanding work which may depend on them has to be allowed to complete. To make things simpler before any of these registers (see behavioral model for a full list) is updated the all the sub units need to be idle (as indicated by the FIFOs linking them be empty).

Please rewrite the sequential paragraphs beginning at line 7 of page 61 as follows:

The main component in the Primary Cache Manager 1070 is the Cache Directory 1102 (one per bank). Block diagrams of this will be given as a significant number of gates are involved in these parts. Note these diagrams only show the major data paths and omit clocks, etc.

The overall block diagram of the Primary Cache Manager 1070 is shown in **Figure 11.**

The cache directory 1102's block diagram is shown in **Figure 12**. Note the complementary key outputs (e.g. K0 and K0\, K1 and K1\) are only used to reduce the cost of the comparators in the CAM cells 1202.

The CAM Cell 1202's block diagram is seen in **Figure 13**. The cache directory can only ever report a maximum of one match per given key.

Please rewrite the paragraph beginning at line 16 of page 64 as follows:

The TLB 1040 holds 16 entries for P3 and 64 entries for RX. The block diagram of the TLB 1040 is seen in Figure 14. The block diagram of an individual CAM cell 1202′ from the TLB 1040 is shown in Figure 15.

Please rewrite the paragraph beginning at line 12 of page 75 as follows:

Figure 16 shows a sample configuration where two rasterizers 1600 are served by a common memory manager and bus interface chip. In the example shown, both chips (1600A and 1600B) have a PCI bus connection to the CPUs as well as an arbitrated connection to memory (through MMU 1610), but of course many other configurations are also possible.

Please insert the following paragraphs after line 15 of page 75:

According to certain disclosed embodiments there is provided: A graphics processing method, comprising the steps of: (a.) caching texture memory fetches using a cache tag assignment which is essentially unique mapped, while (b.) generating condensed cache tags by combining a mip-mapping-level-of-detail parameter which can have at least $2^{J-1}+1$ different values together with coordinate bits which can have as many as 2^K different values into fewer than J+K bits without loss of information (c.) and using said condensed tags for said caching step (a.).

According to certain disclosed embodiments there is provided: A method of generating condensed cache tags, comprising the steps of: (a.) concatenating the texel address on the x- and y-axis with a map level identifier, where addresses on the x-axis can require **m** bits, addresses on the y-axis can require **n** bits, and said map-level identifier can require **p** bits; (b.) if two caches are being used for odd/even maps, deleting the least significant bit of said map level identifier; (c.) if texels are being stored in the cache in $2^i x 2^j$ patches, deleting the **i** least significant bits of the address on the x-axis and the **j** least significant bits of the address on the y-axis; (d.) coding said map level identifier so that the largest map level uses 1 bit to designate the map level and ((**m**-**i**)+(**n**-**j**)) bits to specify said addresses on said x- and y-axis, the second largest map level uses 3 bits to designate the map level and ((**m**-**i**)+(**n**-**j**)-2) bits to specify said addresses on said x-axis and y-axis, and successively smaller map levels use greater than 3 bits to designate the map level and less than ((**m**-**i**)+(**n**-**j**)-2) bits to specify said addresses on said x-axis and y-axis.

According to certain disclosed embodiments there is provided: A cache system for a texture map, comprising: a texture memory containing at least one map, wherein the addresses for said map can require **m** bits for the x-axis, **n** bits for the y-axis, and **p** bits for the map-level identifier; a direct-mapped texture cache for said texture memory wherein a lookup tag requires **m**+**n**-1 or fewer bits.

In the Claims:

1. **(AMENDED)** A graphics processing method, comprising the steps of:
   (a.) caching texture memory fetches, using a cache tag assignment which is essentially unique mapped, while
   (b.) generating condensed cache tags**, corresponding to said cache tag assignment,** by
   combining a mip-mapping-level-of-detail parameter which can have at least $2^{J-1}+1$ different values
   together with
   coordinate bits which can have as many as 2^{K} different values
   into fewer than $J+K$ bits without loss of information
   (c.) and using said condensed tags for said caching step (a.).

2. The method of Claim 1, wherein said step (c) exploits an interrelationship between the number of possible values of said coordinate bits for some values of said mip-mapping-level-of-detail parameter.

3. A graphics processing method, comprising caching texture memory fetches using a cache tag assignment in which a unique relation defines a smaller tag address for any given memory address.

4. The graphics processing method of Claim 3, wherein said cache tag assignment is generated by combining a mip-map-level-of-detail parameter which can have at least $2^{J-1}+1$ different values together with coordinate bits which can have as many as 2^{K} different values into fewer than $J+K$ bits without loss of information.

5. The graphics processing method of Claim 3, wherein said cache tag assignment is generated by combining a first parameter which can have at least $2^{J-1}+1$ different values together with coordinate bits which can have as many as 2^K different values into fewer than $J+K$ bits without loss of information; wherein said first parameter and said coordinate bits are three-dimensional coordinates.

6. **(AMENDED)** A method of generating condensed cache tags, comprising the steps of:

(a.) concatenating the texel address on the x- and y-axis with a map level identifier, where addresses on the x-axis can require m bits, addresses on the y-axis can require n bits, and said map-level identifier can require p bits;

(b.) if two caches are being used for odd/even maps, deleting the least significant bit of said map level identifier;

(c.) if texels are being stored in the cache in $2^i x 2^j$ patches, deleting the i least significant bits of the address on the x-axis and the j least significant bits of the address on the y-axis; and

(d.) coding said map level identifier so that

the largest map level uses 1 bit to designate the map level and ((m-i)+(n-j)) bits to specify said addresses on said x- and y-axis,

the second largest map level uses 3 bits to designate the map level and ((m-i)+(n-j)-2) bits to specify said addresses on said x-axis and y-axis, and

successively smaller map levels use greater than 3 bits to designate the map level and less than ((m-i)+(n-j)-2) bits to specify said addresses on said x-axis and y-axis.

7. **(AMENDED)** A cache system for a texture map, comprising:

a texture memory containing at least one map, wherein the addresses for said map can require m bits for the x-axis, n bits for the y-axis, and p bits for the map-level identifier; and

a direct-mapped texture cache for said texture memory, **configured to be accessed using** ~~wherein~~ ~~a~~ lookup tags **which** requires ~~s~~ m+n-1 or fewer bits.